MNOTES

Make your notes

Gor Yeritsyan

UFAR |IMA2 Paruyr Sevak 1

Abstract

This project presents the development of a Note Management System utilizing Blazor ASP.NET, Microsoft SQL Server, and Entity Framework. The system allows users to create, view, edit, and save notes within a structured environment resembling a file system. The core components of the system include two types of tables: Files and Notes.

Files act as containers similar to folders in a file system, capable of holding both other Files and Notes. Notes are simple entities with attributes such as ID, Parent ID, Name, and Content. The Content attribute of a Note serves as the primary area where users can input, modify, and save their notes.

To facilitate efficient note creation and editing, the project integrates a NuGet extension for a text editor. This editor enhances user experience by providing a user-friendly interface for text manipulation and formatting within the Blazor application.

The integration of Microsoft SQL Server and Entity Framework enables seamless data storage and retrieval, ensuring robustness and scalability of the Note Management System. Users can perform CRUD (Create, Read, Update, Delete) operations on notes with ease, with changes being persisted to the database in real-time.

Contents

Introduction:

In today's digital age, effective note management is crucial for individuals and organizations to stay organized and productive. Traditional methods of note-taking often lack the flexibility and accessibility demanded by modern workflows. Hence, the motivation for this project stems from the need to create a comprehensive digital solution for note management that addresses these shortcomings.

The objectives of the digital note management system are multifaceted. Firstly, it aims to provide users with a user-friendly and intuitive platform for creating, organizing, and accessing notes. Secondly, it seeks to optimize the note-taking process by integrating advanced text editing capabilities, thereby enhancing productivity and collaboration. Additionally, the system aims to ensure seamless data storage and retrieval, enabling users to access their notes anytime, anywhere.

Real-time data collection and analysis play a crucial role in the effectiveness of the digital note management solution. By capturing real-time user interactions, such as note creation, modification, and deletion, the system can provide valuable insights into user behavior and preferences. This data can be leveraged to continually improve the user experience and tailor the system to meet the evolving needs of its users. Furthermore, real-time data analysis enables proactive identification of potential issues or bottlenecks in the system, allowing for timely interventions and optimizations.

In summary, this project aims to address the challenges associated with traditional note-taking methods by developing a robust digital note management solution. By leveraging cutting-edge technologies and real-time data analysis, the system seeks to revolutionize the way individuals and organizations manage their notes, ultimately enhancing productivity and efficiency.

Project Description

First of all, the web notes should be user friendly, for making users to easier understand how to use this program. That’s why it is so minimalistic its only contains crucial parts such as Text Editor and file management window.

This project uses most valuable note taking technique such as CRUD operations (Create, Read, Update, Delete) on notes with ease, with changes persisted to the database.

The Text editor is based on WYSIWYGTextEditor it was imported using NuGet Packages. This editor is based on Javas Script (Quill JS) and has a lot of functionalities such as writing links, changing fonts, you can even provide YouTube link and watch videos inside it. This text editor is better than others, because it can easily save information which is html format that you can place inside the string and also read any data from your provided string. In my opinion it was the best solution for making good notes and provide all the data to Microsoft SQL.

Note this project doesn’t have its own Text Editor because it is time consuming and difficult to make using Blazor and Asp.net. The main logic should be written using JavaScript which is not in our C# course requirements.

Hardware and Software Setup

This project is based on Blazor and Asp.Net framework. It is convenient tool for making web UIs efficiently. However, Blazor hasn't gained widespread industry adoption, unlike JavaScript which remains the industry standard. Because it has its own problems such as it is made for smaller projects. In some cases, it is really slow. Even though it is user friendly and uses most popular OOP langue C# it can’t compete with JavaScript based web frameworks.

This is the list of programs and tools that is used.

Visual Studio 2022 Community Editon

Windows 11

Blazor Web App

Asp.Net

Microsoft SQL 2019

NuGet Packages

MicrosoftEntityFrameWorkCore v 8.02

MicrosoftEntityFrameWorkCore.SqlServer v 8.02

MicrosoftEntityFrameWorkCore.Tools v8.02

SqlTableDependency v8.5.8

WYSIWYGTextEditor v1.01

Aseprite

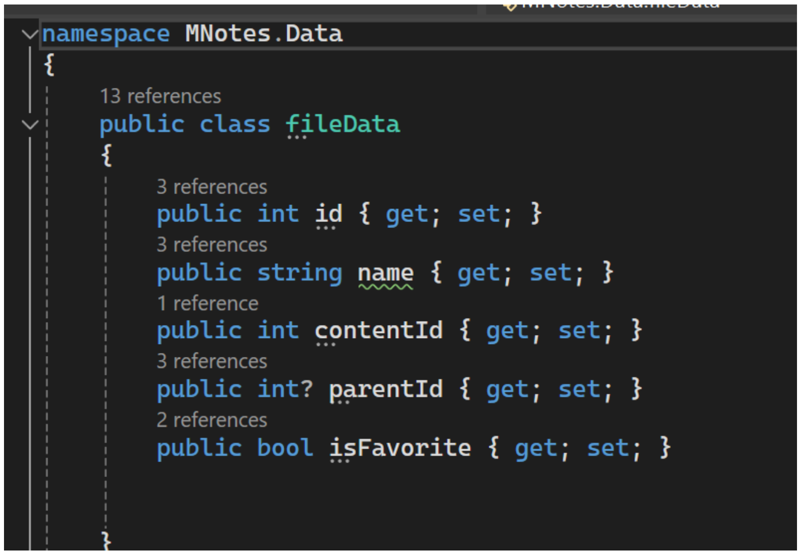
Data Collection

Now let discuss how to collect and save data. In order to do that

1.Make Objects

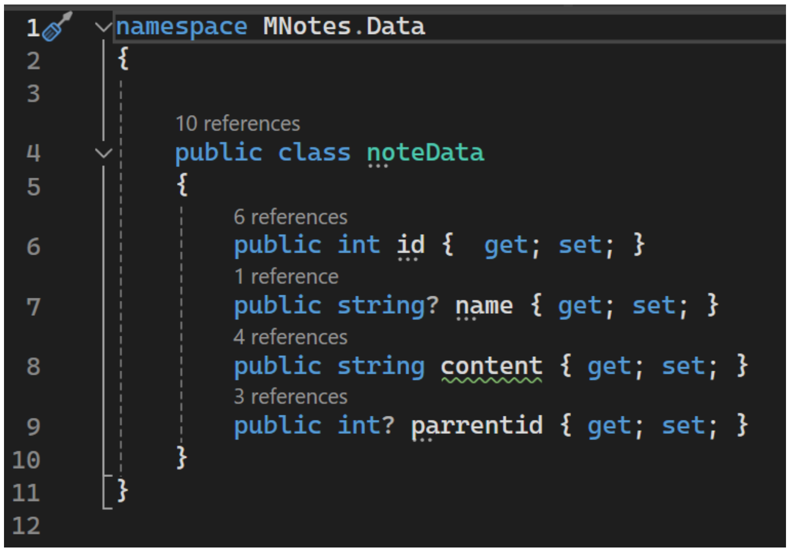
We need to make tables inside Microsoft SQL. Here we have two type of tables Notes and Files.

Files – similar to the folders.



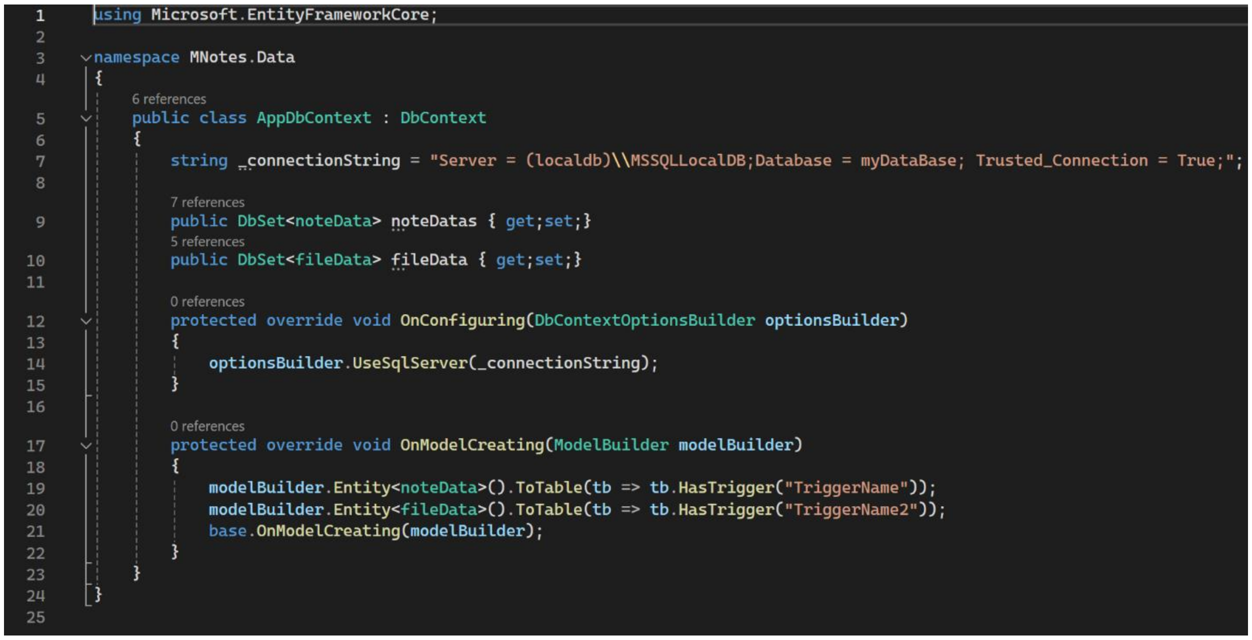
```
namespace MNotes.Data
{
    13 references
    public class fileData
    {
        3 references
        public int id { get; set; }
        3 references
        public string name { get; set; }
        1 reference
        public int contentId { get; set; }
        3 references
        public int? parentId { get; set; }
        2 references
        public bool isFavorite { get; set; }


    }
```

Notes – contains html string in content. id, name and parent id where it should be.



```
namespace MNotes.Data
{

    10 references
    public class noteData
    {
        6 references
        public int id {  get; set; }
        1 reference
        public string? name { get; set; }
        4 references
        public string content { get; set; }
        3 references
        public int? parrentid { get; set; }
    }
}
```

2. Make Application Database Context .cs



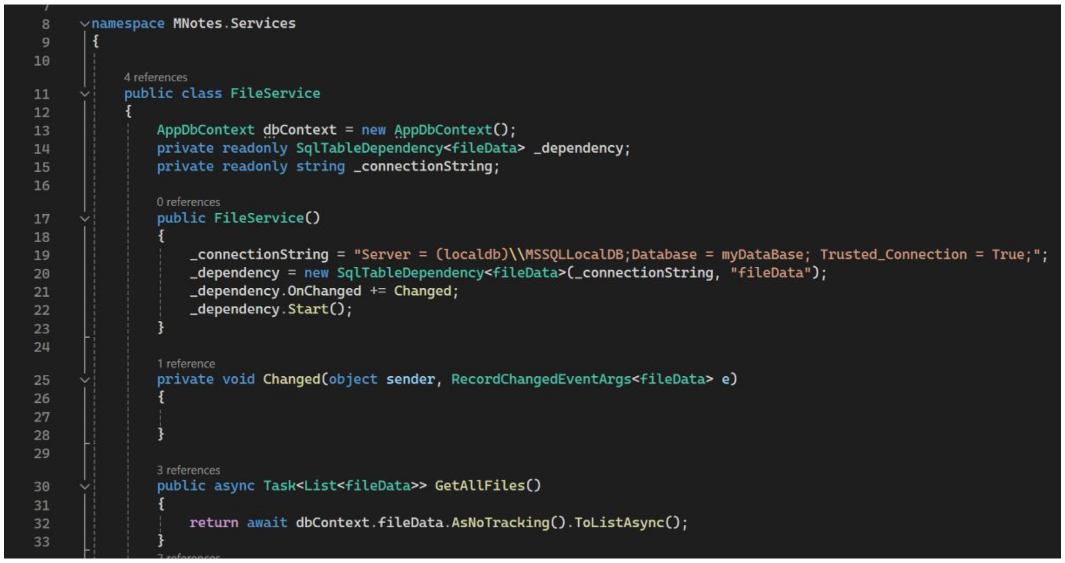
```
1   using Microsoft.EntityFrameworkCore;
2
3   namespace MNotes.Data
4   {
        6 references
5       public class AppDbContext : DbContext
6       {
7           string _connectionString = "Server = (localdb)\\MSSQLLocalDB;Database = myDataBase; Trusted_Connection = True;";
8
            7 references
9           public DbSet<noteData> noteDatas { get;set;}
            5 references
10          public DbSet<fileData> fileData { get;set;}
11
            0 references
12          protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
13          {
14              optionsBuilder.UseSqlServer(_connectionString);
15          }
16
            0 references
17          protected override void OnModelCreating(ModelBuilder modelBuilder)
18          {
19              modelBuilder.Entity<noteData>().ToTable(tb => tb.HasTrigger("TriggerName"));
20              modelBuilder.Entity<fileData>().ToTable(tb => tb.HasTrigger("TriggerName2"));
21              base.OnModelCreating(modelBuilder);
22          }
23      }
24  }
25
```

We use DbSet it is similar to the lists and collect all the noteDatas and fileDatas , but inorder to do that wee need to connect to our MS SQL database .In line 7 this is the connection string that contains local server name Database Name and Trusted connection property.

On Configuring we connect to our SQL Server and if everything is alright our tables will appear in our DbSet lists.

Also, we have onModelCreating we make database triggers because it said that there was a trigger inside our tables but we didn't do anything. That's why I had to add these two lines in (19,20) to create our triggers without any limitation with accessing data.

3. Make Service



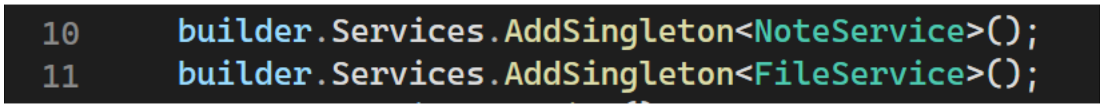
```
namespace MNotes.Services
{

    4 references
    public class FileService
    {
        AppDbContext dbContext = new AppDbContext();
        private readonly SqlTableDependency<fileData> _dependency;
        private readonly string _connectionString;

        0 references
        public FileService()
        {
            _connectionString = "Server = (localdb)\\MSSQLLocalDB;Database = myDataBase; Trusted_Connection = True;";
            _dependency = new SqlTableDependency<fileData>(_connectionString, "fileData");
            _dependency.OnChanged += Changed;
            _dependency.Start();
        }

        1 reference
        private void Changed(object sender, RecordChangedEventArgs<fileData> e)
        {

        }

        3 references
        public async Task<List<fileData>> GetAllFiles()
        {
            return await dbContext.fileData.AsNoTracking().ToListAsync();
        }
```

This is fileServices we need this in order to access to our files and get or edit it.

For example, in Line 30 we have function that gets all filesData from dbContext

dbConext is the previous file that we have created in section two.

We have also same thing for our Notes. Don’t forget to add NoteServices to the Program.cs file.

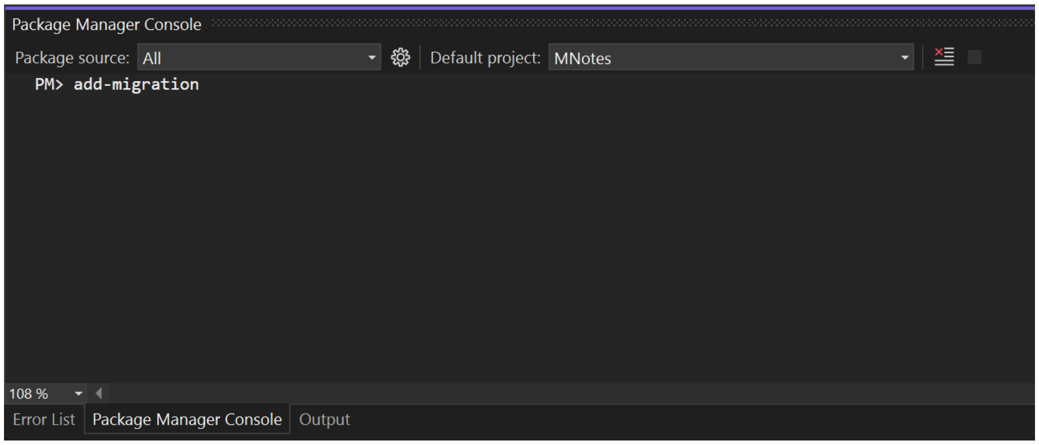


```
builder.Services.AddSingleton<NoteService>();
builder.Services.AddSingleton<FileService>();
```

4. Create migration

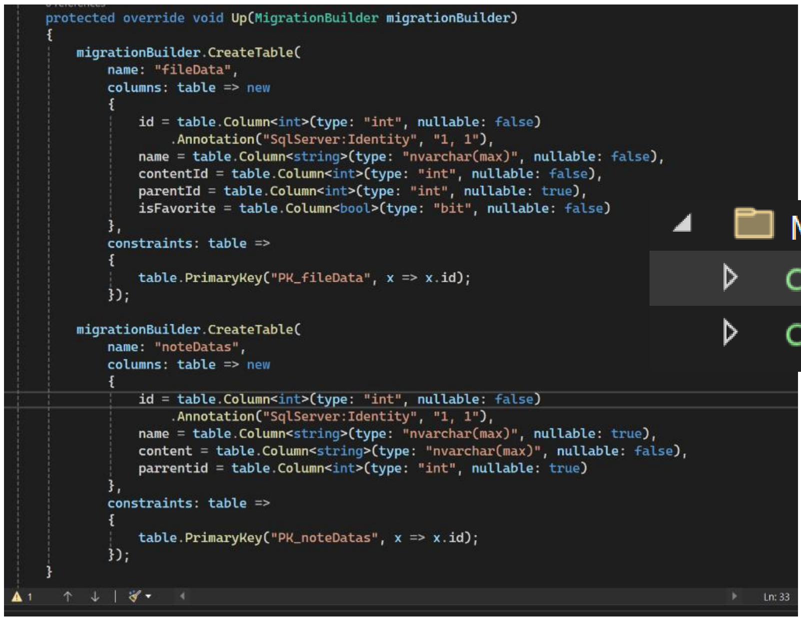
We need to open Package Manager Console

And type this command add-migration (name)

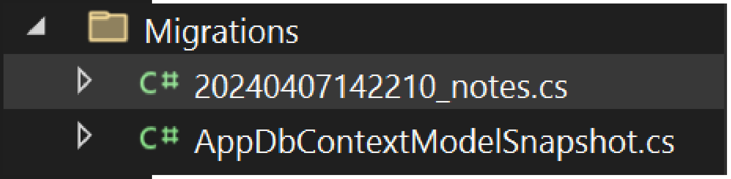


```
PM> add-migration
```

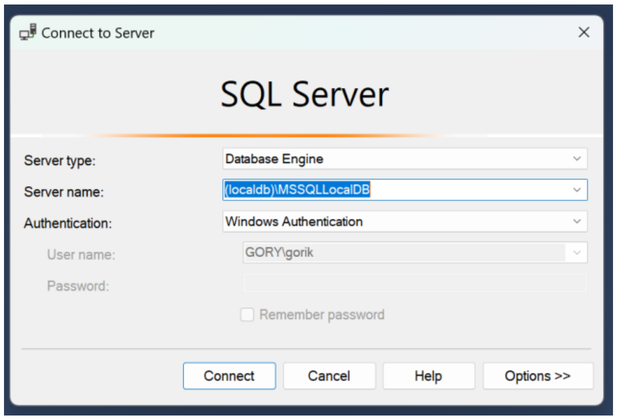
After successfully adding migration we need to update our database



```
protected override void Up(MigrationBuilder migrationBuilder)
{
    migrationBuilder.CreateTable(
        name: "fileData",
        columns: table => new
        {
            id = table.Column<int>(type: "int", nullable: false)
                .Annotation("SqlServer:Identity", "1, 1"),
            name = table.Column<string>(type: "nvarchar(max)", nullable: false),
            contentId = table.Column<int>(type: "int", nullable: false),
            parentId = table.Column<int>(type: "int", nullable: true),
            isFavorite = table.Column<bool>(type: "bit", nullable: false)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_fileData", x => x.id);
        });

    migrationBuilder.CreateTable(
        name: "noteDatas",
        columns: table => new
        {
            id = table.Column<int>(type: "int", nullable: false)
                .Annotation("SqlServer:Identity", "1, 1"),
            name = table.Column<string>(type: "nvarchar(max)", nullable: true),
            content = table.Column<string>(type: "nvarchar(max)", nullable: false),
            parrentid = table.Column<int>(type: "int", nullable: true)
        },
        constraints: table =>
        {
            table.PrimaryKey("PK_noteDatas", x => x.id);
        });
}
```



5. Open MS SQL

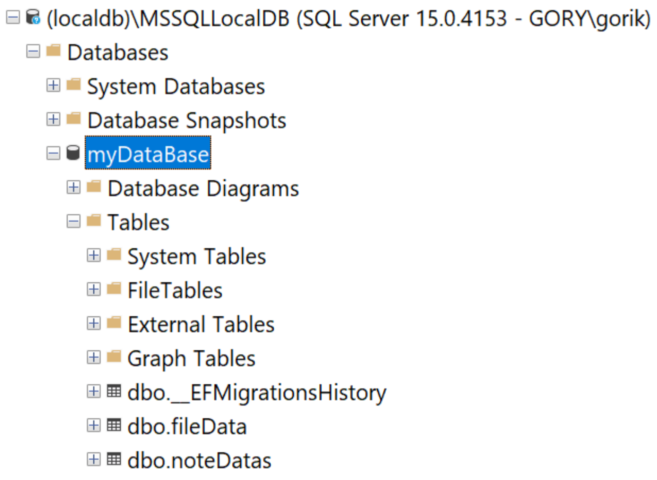


Connect to Server

SQL Server

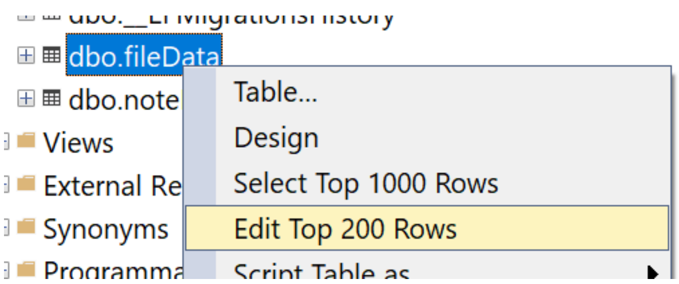
Server type: Database Engine
Server name: (localdb)\MSSQLLocalDB
Authentication: Windows Authentication
User name: GORY\gorik
Password:
Remember password

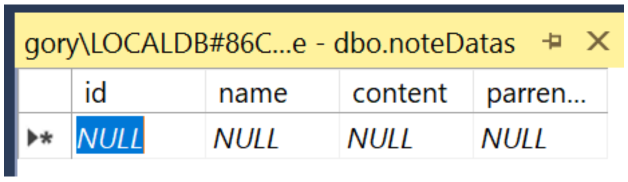
I have already made my database and made a migration



```
(localdb)\MSSQLLocalDB (SQL Server 15.0.4153 - GORY\gorik)
  Databases
    System Databases
    Database Snapshots
    myDataBase
      Database Diagrams
      Tables
        System Tables
        FileTables
        External Tables
        Graph Tables
        dbo.__EFMigrationsHistory
        dbo.fileData
        dbo.noteDatas
```

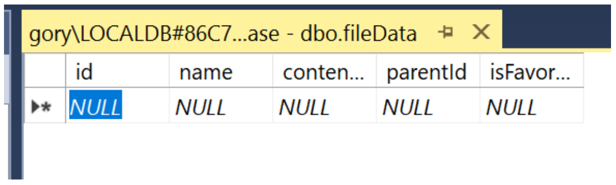
Now as you can see inside myDataBase appeared Tables folder with 3 new files. Migration history that shows bassicly all the manipulations with your database. FileData table and noteDatas table.





gory\LOCALDB#86C...e - dbo.noteDatas

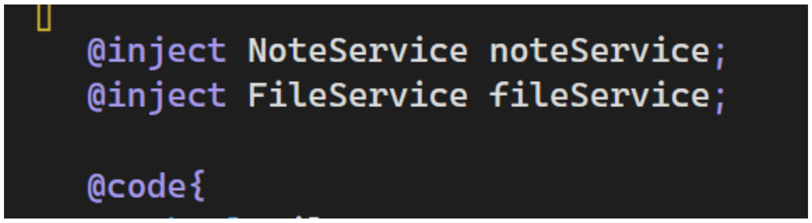
	id	name	content	parren...
▶*	NULL	NULL	NULL	NULL



gory\LOCALDB#86C7...ase - dbo.fileData

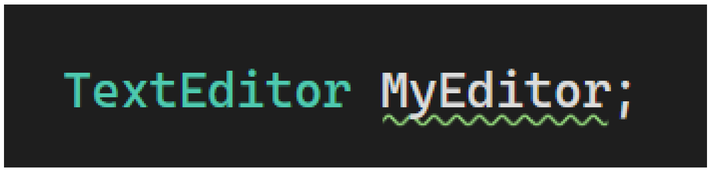
	id	name	conten...	parentId	isFavor...
▶*	NULL	NULL	NULL	NULL	NULL

6. Home.razor file
Inside razor file we inject our Services and after that get data from it.



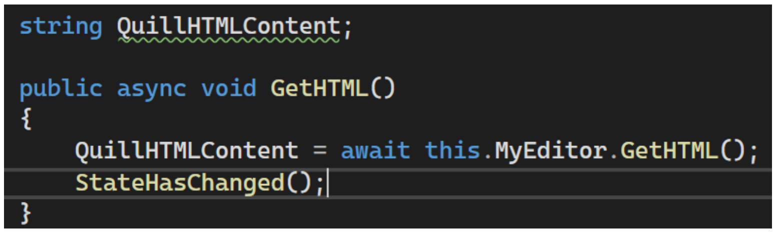
```
@inject NoteService noteService;
@inject FileService fileService;

@code{
```

We create our Text editor.



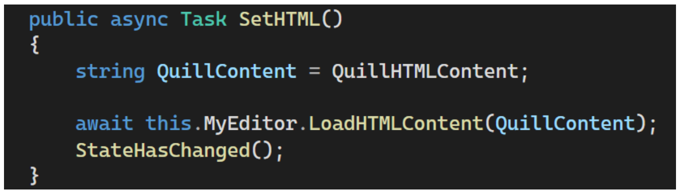
```
TextEditor MyEditor;
```

Here is a function example we give our QuillHtmlContent string our editor content.



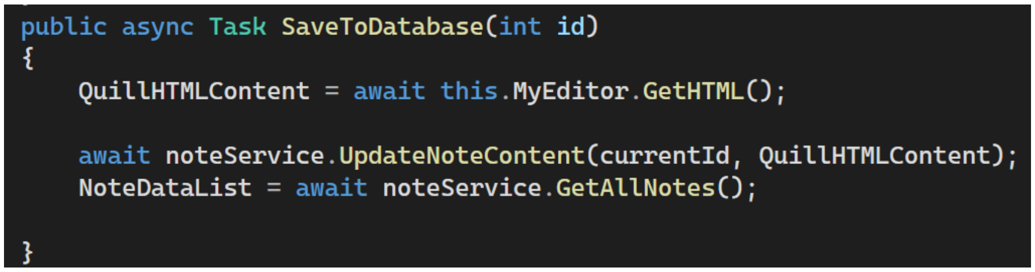
```
string QuillHTMLContent;

public async void GetHTML()
{
    QuillHTMLContent = await this.MyEditor.GetHTML();
    StateHasChanged();
}
```

Same thing with making our editor to show our text.



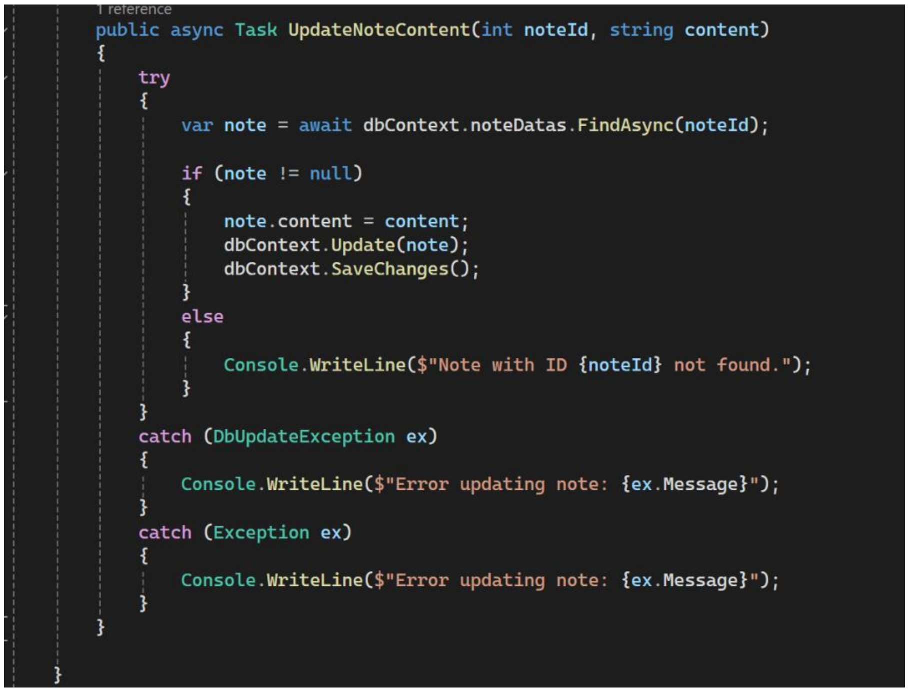
```
public async Task SetHTML()
{
    string QuillContent = QuillHTMLContent;

    await this.MyEditor.LoadHTMLContent(QuillContent);
    StateHasChanged();
}
```

But now how to save to database



```
public async Task SaveToDatabase(int id)
{
    QuillHTMLContent = await this.MyEditor.GetHTML();

    await noteService.UpdateNoteContent(currentId, QuillHTMLContent);
    NoteDataList = await noteService.GetAllNotes();

}
```

In our case we give id and html string to our noteService to update it.



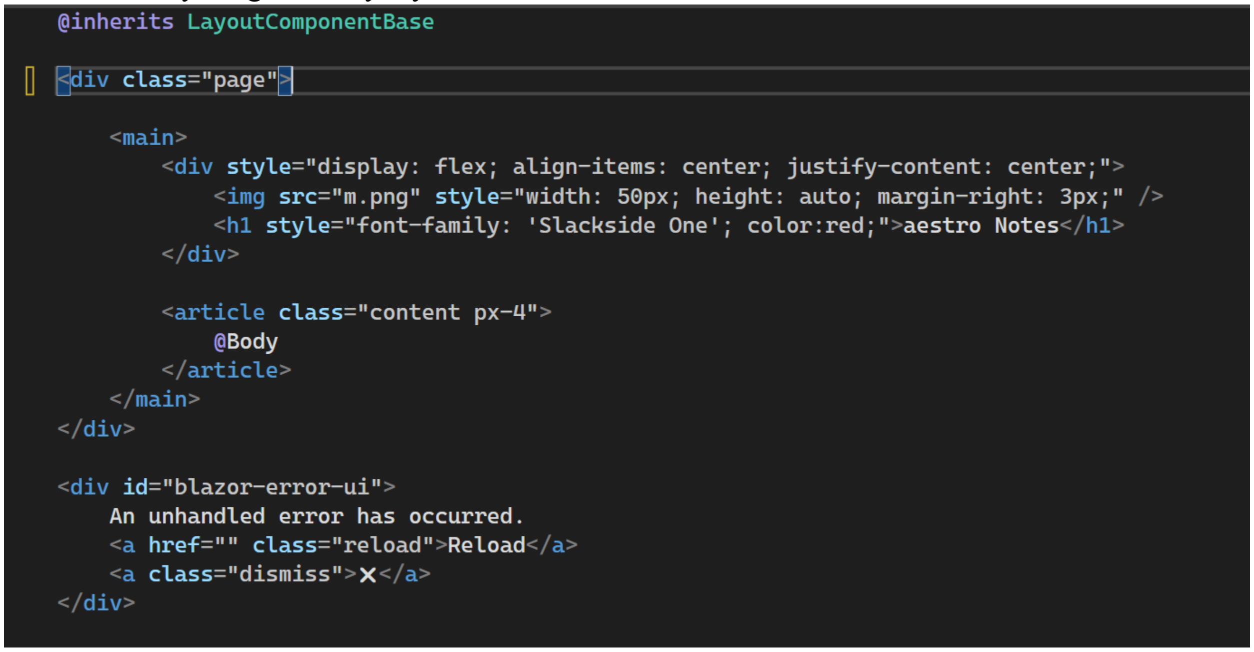
```
        1 reference
        public async Task UpdateNoteContent(int noteId, string content)
        {
            try
            {
                var note = await dbContext.noteDatas.FindAsync(noteId);

                if (note != null)
                {
                    note.content = content;
                    dbContext.Update(note);
                    dbContext.SaveChanges();
                }
                else
                {
                    Console.WriteLine($"Note with ID {noteId} not found.");
                }
            }
            catch (DbUpdateException ex)
            {
                Console.WriteLine($"Error updating note: {ex.Message}");
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error updating note: {ex.Message}");
            }
        }

    }
```

We try and catch all the errors. In our case it works and changes and updates inside database.

Main Page

This part will be about “Home.razor” file and page layout.

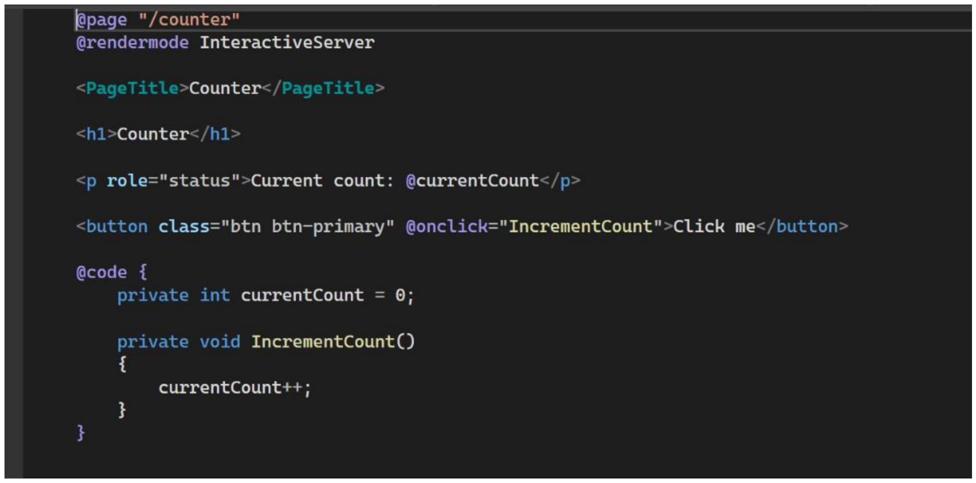
In blazor everything made by layout



```
@inherits LayoutComponentBase

<div class="page">

    <main>
        <div style="display: flex; align-items: center; justify-content: center;">
            <img src="m.png" style="width: 50px; height: auto; margin-right: 3px;" />
            <h1 style="font-family: 'Slackside One'; color:red;">aestro Notes</h1>
        </div>

        <article class="content px-4">
            @Body
        </article>
    </main>
</div>

<div id="blazor-error-ui">
    An unhandled error has occurred.
    <a href="" class="reload">Reload</a>
    <a class="dismiss">✕</a>
</div>
```

Inside @Body executes our pages. We also can have Navigation Menu but in this case I disabled it.

In pages folder we have our .razor files our main program works in Home.razor in previous versions it was index.razor.

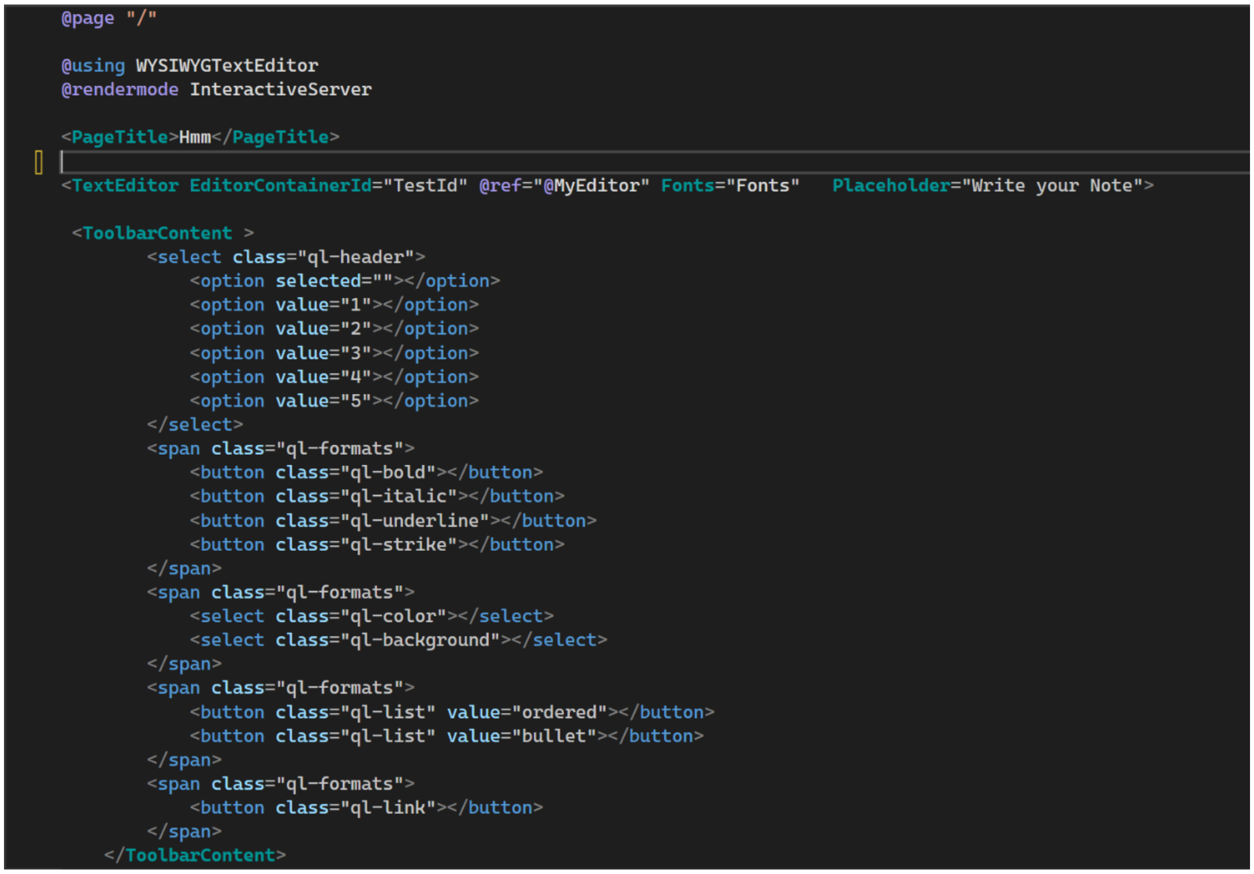


```
@page "/counter"
@rendermode InteractiveServer

<PageTitle>Counter</PageTitle>

<h1>Counter</h1>

<p role="status">Current count: @currentCount</p>

<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

@code {
    private int currentCount = 0;

    private void IncrementCount()
    {
        currentCount++;
    }
}
```

This is some default examples. Counter where @page “is a link if it is default it is like this / ”

@rendermode InteractiveServer is necessary for buttons and other dynamic interactions.

After that we write our html if we want to give some values we can use @ this sign means code part. Inside @code we can write our C# functions and variables.

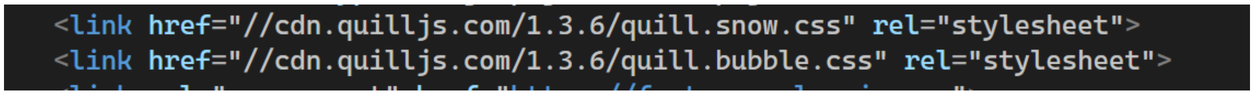


```
@page "/"

@using WYSIWYGTextEditor
@rendermode InteractiveServer

<PageTitle>Hmm</PageTitle>

<TextEditor EditorContainerId="TestId" @ref="@MyEditor" Fonts="Fonts"   Placeholder="Write your Note">

 <ToolbarContent >
        <select class="ql-header">
            <option selected=""></option>
            <option value="1"></option>
            <option value="2"></option>
            <option value="3"></option>
            <option value="4"></option>
            <option value="5"></option>
        </select>
        <span class="ql-formats">
            <button class="ql-bold"></button>
            <button class="ql-italic"></button>
            <button class="ql-underline"></button>
            <button class="ql-strike"></button>
        </span>
        <span class="ql-formats">
            <select class="ql-color"></select>
            <select class="ql-background"></select>
        </span>
        <span class="ql-formats">
            <button class="ql-list" value="ordered"></button>
            <button class="ql-list" value="bullet"></button>
        </span>
        <span class="ql-formats">
            <button class="ql-link"></button>
        </span>
    </ToolbarContent>
```

This is my main page Text editor that I import by @using

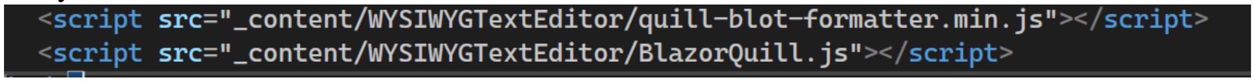
However using won't work without adding proper NuGet file and add inside App.razor

Head



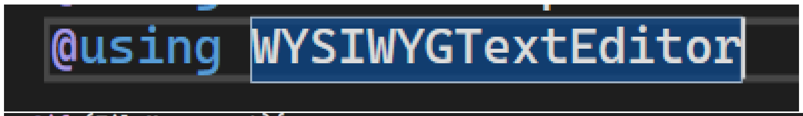
```
<link href="//cdn.quilljs.com/1.3.6/quill.snow.css" rel="stylesheet">
<link href="//cdn.quilljs.com/1.3.6/quill.bubble.css" rel="stylesheet">
```

Body

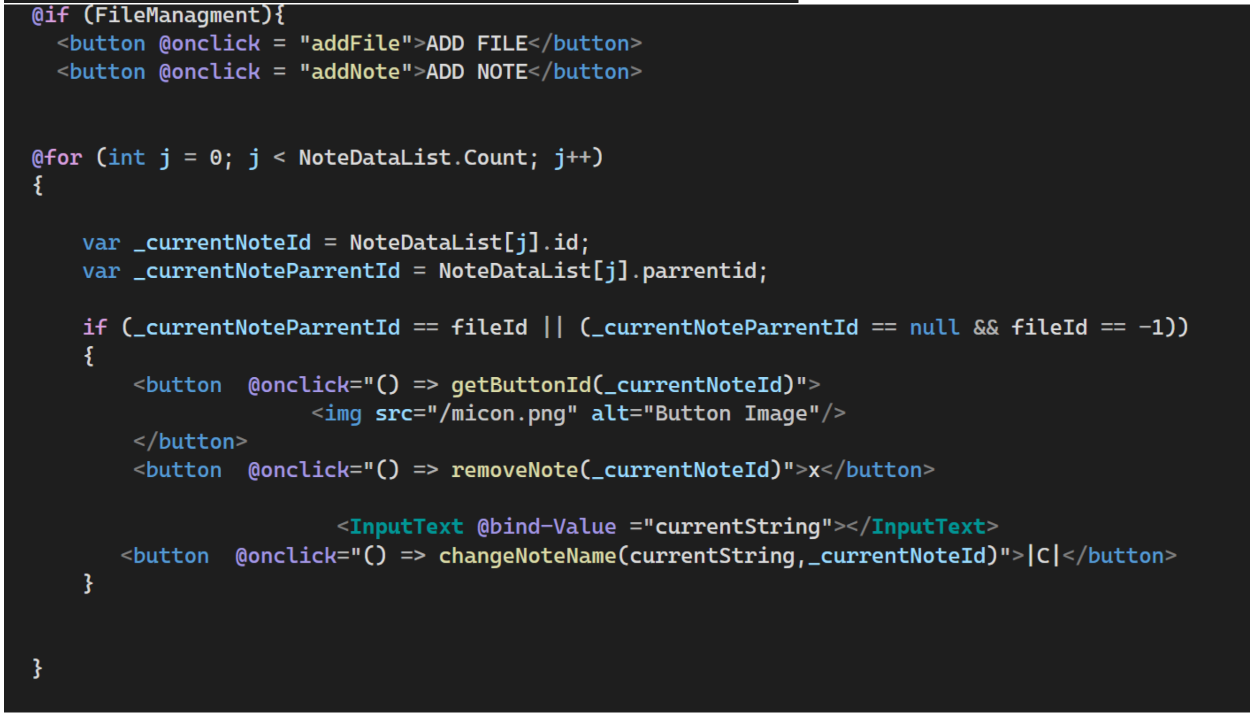


```
<script src="_content/WYSIWYGTextEditor/quill-blot-formatter.min.js"></script>
<script src="_content/WYSIWYGTextEditor/BlazorQuill.js"></script>
```

and lastly add into _Imports.razor.



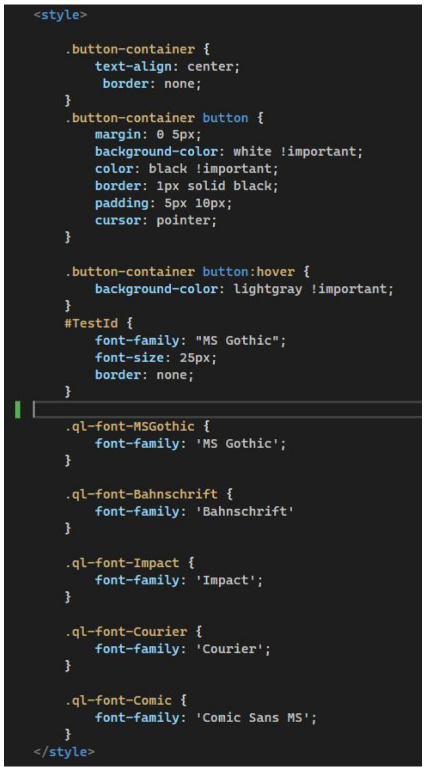
```
@using WYSIWYGTextEditor
```



```
@if (FileManagment){
  <button @onclick = "addFile">ADD FILE</button>
  <button @onclick = "addNote">ADD NOTE</button>


@for (int j = 0; j < NoteDataList.Count; j++)
{

    var _currentNoteId = NoteDataList[j].id;
    var _currentNoteParrentId = NoteDataList[j].parrentid;

    if (_currentNoteParrentId == fileId || (_currentNoteParrentId == null && fileId == -1))
    {
        <button  @onclick="() => getButtonId(_currentNoteId)">
                      <img src="/micon.png" alt="Button Image"/>
        </button>
        <button  @onclick="() => removeNote(_currentNoteId)">x</button>

                        <InputText @bind-Value ="currentString"></InputText>
       <button  @onclick="() => changeNoteName(currentString,_currentNoteId)">|C|</button>
    }


}
```

Here is some example of showing Notes using html with code. That’s why blazor is interesting the whole page renders dynamically.

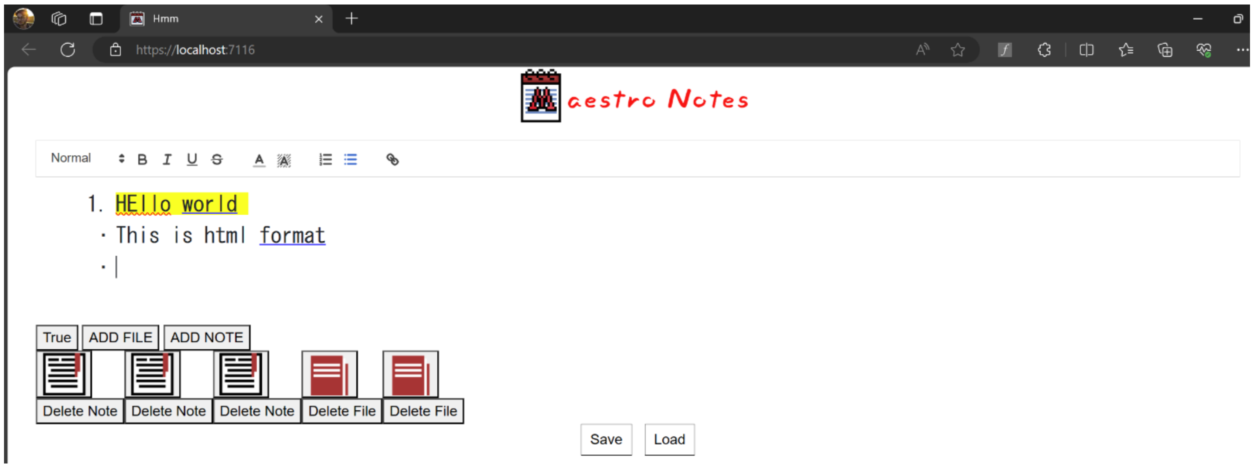
Also we can write css inside razor file .For example our div layout . Button style. Or as you can see custom Font styles for our TextEditor.



```
<style>

    .button-container {
        text-align: center;
         border: none;
    }
    .button-container button {
        margin: 0 5px;
        background-color: white !important;
        color: black !important;
        border: 1px solid black;
        padding: 5px 10px;
        cursor: pointer;
    }

    .button-container button:hover {
        background-color: lightgray !important;
    }
    #TestId {
        font-family: "MS Gothic";
        font-size: 25px;
        border: none;
    }

    .ql-font-MSGothic {
        font-family: 'MS Gothic';
    }

    .ql-font-Bahnschrift {
        font-family: 'Bahnschrift'
    }

    .ql-font-Impact {
        font-family: 'Impact';
    }

    .ql-font-Courier {
        font-family: 'Courier';
    }

    .ql-font-Comic {
        font-family: 'Comic Sans MS';
    }
</style>
```

Testing and Validation

For testing we run program and opens the localhost: with 7116 port



1. HEllo world
- This is html format

Here you can write your note. Below as you may see is the file management. Where you can easily add either not or file. File can contain notes and files. Note just displays html string with its id.

Under that it is Save and Load buttons. Save for saving your current progress to the last chosen file. Load is get data from last chosen file. You can use it as undo if you didn't save your progress.

The problems and solutions

Fist problems was when I first made my migration. It couldn't take any values from tables because it had some triggers. I did a little bit research and find out that I should add two lines to dbcontext.cs file in order to fix that problem.

Second problems is the how blazor sometimes crashes. It is hard to manipulate with blazor a lot of operations at once. This is not uniquely blazor problem it is web development problem because it can't do multithreading operations, it means divide task into the smaller tasks and do simultaneously. For example, when you press 4-5 times add note it could crash. The solution is add some delay after operations. Which fixes the problem but sometimes you can't avoid random crashes.

And lastly Inside this Text Editor exists image import, however when you import image it couldn't understand the path or give some value to save it to database. One of the solutions was to find all the images from my html string and convert it into the Base 64 format. It is also like a string uses characters and numbers but the problem was that this editor that import image via folder can't take track of it. The second problem is the place it should show the image because in our html you could place your image at the very top or wherever you want. How can we know exactly the position of it?

Future Enhancements

In this project we can think a lot of interesting additions but they will be time consuming and complicate the initial concept of being minimalistic.

Design

The design is important part for user experience. In my opinion it needs improvement for example to make drop down menu with all the necessary documents. Also changing folders (files) and notes design color style, background colors, pictures.

Autosave

Make autosaves. In this application saves done from user it may be more convenient to make saves automatically. But the problem is it is resource consuming. Which will again be difficult to handle with blazor application, or making it after pressing enter which is similar to my solution with button.

Bookmarks

Bookmarks for notes really convenient thing that will fix file management mess problems.

Image importing

Importing and saving images are hard but also convenient thing to make better experience and useful feature.

Music

The initial idea was to add music player and play songs.

AI correcting and suggestion

Using chatgpt 4 or Grammarly in order to help user to correct/suggest writing correct. Also, code writing.

Overall, this project was made for personal use and the features mentioned above will probably added and polished.

Conclusion

In conclusion, the development of a digital note management system using Blazor ASP.NET, Microsoft SQL Server, and Entity Framework represents a significant step towards offering a modern, efficient, and user-friendly solution for note-taking and organization. Despite JavaScript's dominance in the industry, the project opted for Blazor ASP.NET to leverage its advantages within the .NET ecosystem.

Through the integration of Blazor, the project demonstrates the potential of this emerging technology stack to streamline development processes, enhance performance, and leverage existing .NET expertise. By providing a platform for creating, editing, and organizing notes within a structured environment, the system aims to improve productivity and efficiency for individuals and organizations alike.

Furthermore, the integration of real-time data collection and analysis capabilities enables continuous improvement and optimization of the system, ensuring it remains responsive to user needs and preferences. The seamless integration with Microsoft SQL Server and Entity Framework ensures robust data storage and retrieval, enabling users to access their notes anytime, anywhere.

Overall, the digital note management system underscores the project's commitment to exploring innovative solutions within the .NET ecosystem while addressing the evolving needs of modern workflows. By embracing Blazor ASP.NET and leveraging its capabilities, the project sets a precedent for developing efficient and scalable web applications that prioritize user experience and productivity.

References

Useful references.

1. Github Link

2. Blazor YouTube tutorial

3. Blazor Documentation

4. Learn C#

5. Text Editor Github

6. Blazor Book

7. ASP.NET Book

9. Microsoft SQL Documentation